This listing of claims will replace all prior versions, and listings, of claims in the application:

<u>Listing of Claims:</u>

1   1.   (Currently Amended) A method for executing a commit instruction
2   to facilitate transactional execution on a processor, comprising:
3   <u>executing a block of instructions transactionally, wherein executing the</u>
4   <u>block of instructions transactionally involves placing load-marks on cache lines</u>
5   <u>from which data is loaded and placing store-marks on cache lines to which data is</u>
6   <u>stored;</u>
7   encountering the commit instruction during execution of a program,
8   wherein the commit instruction marks the end of a block of instructions to be
9   executed transactionally; and
10   upon encountering the commit instruction, successfully completing
11   transactional execution of the block of instructions preceding the commit
12   instruction, wherein successfully completing the transactional execution involves
13   atomically committing changes made during the transactional execution by:
14   treating store-marked cache lines as locked, thereby causing other
15   processes to wait to access the store-marked cache lines;
16   committing store buffer entries generated during transactional
17   execution to memory, wherein committing each store buffer entry involves
18   removing the store-mark from, and thereby unlocking, a corresponding
19   store-marked cache line;
20   clearing load-marks from cache lines; and

2

21   committing register file changes made during transactional

22   execution;

23   wherein changes made during the transactional execution are not

24   committed to the architectural state of the processor until the transactional

25   execution successfully completes.

1   2.   (Previously Presented) The method of claim 1, wherein

2   successfully completing the transactional execution involves:

3   resuming normal non-transactional execution.

1   3.   (Cancelled)

1   4.   (Original) The method of claim 1, wherein if an interfering data

2   access from another process is encountered during the transactional execution and

3   prior to encountering the commit instruction, the method further comprises:

4   discarding changes made during the transactional execution; and

5   attempting to re-execute the block of instructions.

1   5.   (Previously Presented) The method of claim 1, wherein for a

2   variation of the commit instruction, successfully completing the transactional

3   execution involves:

4   commencing transactional execution of the block of instructions following

5   the commit instruction.

1   6.   (Original) The method of claim 1, wherein potentially interfering

2   data accesses from other processes are allowed to proceed during the transactional

3   execution of the block of instructions.

1   7.   (Original) The method of claim 1, wherein the block of instructions
2   to be executed transactionally comprises a critical section.

1   8.   (Original) The method of claim 1, wherein the commit instruction
2   is a native machine code instruction of the processor.

1   9.   (Original) The method of claim 1, wherein the commit instruction
2   is defined in a platform-independent programming language.

1   10.   (Currently Amended) A computer system that supports a commit
2   instruction to facilitate transactional execution, wherein the commit instruction
3   marks the end of a block of instructions to be executed transactionally,
4   comprising:
5   a processor; and
6   an execution mechanism within the processor, wherein the execution
7   mechanism is configured to place load-marks on cache lines from which data is
8   loaded and place store-marks on cache lines to which data is stored during
9   transactional execution;
10   wherein upon encountering the commit instruction, the execution
11   mechanism is configured to successfully complete transactional execution of the
12   block of instructions preceding the commit instruction, wherein successfully
13   completing the transactional execution involves atomically committing changes
14   made during the transactional execution by:
15   treating store-marked cache lines as locked, thereby causing other
16   processes to wait to access the store-marked cache lines;

4

17   committing store buffer entries generated during transactional
18   execution to memory, wherein committing each store buffer entry involves
19   removing the store-mark from, and thereby unlocking, a corresponding
20   store-marked cache line;
21   clearing load-marks from cache lines; and
22   committing register file changes made during transactional
23   execution;
24   wherein changes made during the transactional execution are not
25 committed to the architectural state of the processor until the transactional
26 execution successfully completes.

1   11.   (Previously Presented) The computer system of claim 10, wherein
2   while successfully completing transactional execution, the execution mechanism
3   is configured to:
4   resume normal non-transactional execution.

1   12.   (Cancelled)

1   13.   (Original) The computer system of claim 10, wherein if an
2   interfering data access from another process is encountered during the
3   transactional execution and prior to encountering the commit instruction, the
4   execution mechanism is configured to:
5   discard changes made during the transactional execution; and to
6   attempt to re-execute the block of instructions.

14. (Previously Presented) The computer system of claim 10, wherein if a variation of the commit instruction is encountered, the execution mechanism is configured to:

commence transactional execution of the block of instructions following the commit instruction.

15. (Original) The computer system of claim 10, wherein the computer system is configured to allow potentially interfering data accesses from other processes to proceed during the transactional execution of the block of instructions.

16. (Original) The computer system of claim 10, wherein the block of instructions to be executed transactionally comprises a critical section.

17. (Original) The computer system of claim 10, wherein the commit instruction is a native machine code instruction of the processor.

18. (Original) The computer system of claim 10, wherein the commit instruction is defined in a platform-independent programming language.

19. (Currently Amended) A computer-readable storage medium storing instructions that when executed by a computer cause the computer to perform a method for executing a commit instruction to facilitate transactional execution, comprising:

executing a block of instructions transactionally, wherein executing the block of instructions transactionally involves placing load-marks on cache lines

6

7 from which data is loaded and placing store-marks on cache lines to which data is

8 stored;

9   encountering the commit instruction during execution of a program,

10 wherein the commit instruction marks the end of a block of instructions to be

11 executed transactionally; and

12   upon encountering the commit instruction, successfully completing

13 transactional execution of the block of instructions preceding the commit

14 instruction, wherein successfully completing the transactional execution involves

15 atomically committing changes made during the transactional execution by:

16   treating store-marked cache lines as locked, thereby causing other

17   processes to wait to access the store-marked cache lines;

18   committing store buffer entries generated during transactional

19   execution to memory, wherein committing each store buffer entry involves

20   removing the store-mark from, and thereby unlocking, a corresponding

21   store-marked cache line;

22   clearing load-marks from cache lines; and

23   committing register file changes made during transactional

24   execution;

25   wherein changes made during the transactional execution are not

26 committed to the architectural state of the processor until the transactional

27 execution successfully completes.

1   20.   (Previously Presented) The computer-readable storage medium of

2 claim 19, wherein successfully completing transactional execution involves:

3   resuming normal non-transactional execution.

7